C# 프로그래밍

- 15 주차 (1강)

SEJONG UNIVERSITY

대리자 체인
익명 메서드
이벤트

SEJONG UNIVERSITY

대리자 인스턴스의 메소드 참조

- C# 2.0부터는 대리자 인스턴스에 메소드를 쉽게 연결하도록 지원
  - ✓ 기존에는 new 연산자를 사용하고 생성자의 인수로 메소드 이름을 입력
  - ✓ C# 2.0부터는 '=' 연산자를 사용해 대리자 인스턴스에 메소드를 직접 대입 가능

사용 예제:

```
delegate int MyDelegate(int a, int b);

static int Plus(int a, int b)
{
    return a + b;
}

Static void Main(string[] args)
{
    MyDelegate Callback1, Callback2;
    Callback1 = new MyDelegate(Plus);
    Callback2 = Plus; // new 없이 마치 값을 대입하듯이 사용 가능

    Console.WriteLine($"{Callback1(3,4)} {Callback2(3, 4)}"); // 출력: 7 7

}
```

대리자 체인

- 대리자 하나가 여러 개의 메서드를 동시에 참조
  - ✓ '+=' 연산자를 이용하여 대리자 인스턴스에 여러 메서드를 결합 가능

사용 예제: 대리자와 대리자 형식에 맞춘 세개의 메서드

```
delegate void TherelsAFire(string location);

static void Call119 (string location)
{
    Console.WriteLine("소방서죠? 불났어요! 주소는 {0}", location);
}

static void ShotOut(string location)
{
    Console.WriteLine("피하세요! {0}에 불이 났어요!", location);
}

static void Escape(string location)
{
    Console.WriteLine("{0}에서 나갑시다!", location);
}
```

대리자 인스턴스가 모든 메서드를 참조하도록 '+=' 연산자를 이용하여 결합

```
static void Main(string[] args)
{
    TherelsAFire Fire = new TherelsAFire(Call119);
    Fire += new TherelsAFire(ShotOut);
    //Fire += new TherelsAFire(Escape);
    Fire += Escape;

    Fire("우리집");
}
```

대리자 체인을 따라 차례대로 호출

출력 결과:
소방서죠? 불났어요! 주소는 우리집
피하세요! 우리집에 불이 났어요!
우리집에서 나갑시다!

대리자 체인

- 대리자 하나가 여러 개의 메서드를 동시에 참조
  - ✓ '+=' 연산자가 아니어도 다음의 방법들로 대리자 체인 생성 가능

```
TherelsAFire Fire = new TherelsAFire(Call119)
    + new TherelsAFire(ShotOut)
    + new TherelsAFire(Escape);
```

```
TherelsAFire Fire = (TherelsAFire)Delegate.Combine(
    new TherelsAFire(Call119),
    new TherelsAFire(ShotOut),
    new TherelsAFire(Escape));
```

  - ✓ 대리자 체인에서 특정 대리자를 끊어낼 때는 '-=' 연산자를 이용

```
TherelsAFire Fire = new TherelsAFire(Call119)
    + new TherelsAFire(ShotOut)
    + new TherelsAFire(Escape);

Fire -= new TherelsAFire(ShotOut);
Fire -= Escape;
Fire("우리집"); // Call119만 출력: 소방서죠? 불났어요! 주소는 우리집
```

대리자 체인 예제코드

```
delegate void Notify(string message);

class Notifier
{
    public Notify EventOccured; // Notify 대리자 선언
}

class EventListener // Notify 대리자의 인스턴스
{           // EventOccured를 가지는 클래스 Notifier 선언
    private string name;
    public EventListener(string name)
    {
        this.name = name;
    }

    public void SomethingHappend(string message)
    {
        Console.WriteLine($"{name}.SomethingHappend : {message}");
    }
}
```

```
static void Main(string[] args)
{
    Notifier notifier = new Notifier();
    EventListener listener1 = new EventListener("Listener1");
    EventListener listener2 = new EventListener("Listener2");
    EventListener listener3 = new EventListener("Listener3");
    // +=연산자를 이용한 체인 만들기
    notifier.EventOccured += listener1.SomethingHappend;
    notifier.EventOccured += listener2.SomethingHappend;
    notifier.EventOccured += listener3.SomethingHappend;
    notifier.EventOccured("You've got mail");

    Console.WriteLine();
    // -=연산자를 이용한 체인 끊기
    notifier.EventOccured -= listener2.SomethingHappend;
    notifier.EventOccured("Download complete.");
}
```

출력 결과:

```
Listener1.SomethingHappend : You've got mail
Listener2.SomethingHappend : You've got mail
Listener3.SomethingHappend : You've got mail

Listener1.SomethingHappend : Download complete.
Listener3.SomethingHappend : Download complete.
```

대리자 체인 예제코드

```
delegate void Notify(string message);

class Notifier
{
    public Notify EventOccured;
}

class EventListener
{
    private string name;
    public EventListener(string name)
    {
        this.name = name;
    }

    public void SomethingHappend(string message)
    {
        Console.WriteLine($"{name}.SomethingHappend : {message}");
    }
}
```

```
static void Main(string[] args)
{   // +,= 연산자를 이용한 체인 만들기
    notifier.EventOccured = new Notify(listener2.SomethingHappend)
        + new Notify(listener3.SomethingHappend);
    notifier.EventOccured("Nuclear launch detected.");

    Console.WriteLine();

    Notify notify1 = new Notify(listener1.SomethingHappend);
    Notify notify2 = new Notify(listener2.SomethingHappend);
    // Delegate.Combine() 메서드를 이용한 체인 만들기
    notifier.EventOccured = (Notify)Delegate.Combine(notify1, notify2);
    notifier.EventOccured("Fire!!");

    Console.WriteLine();
    // Delegate.Remove() 메서드를 이용한 체인 끊기
    notifier.EventOccured =
        (Notify)Delegate.Remove(notifier.EventOccured, notify2);
    notifier.EventOccured("RPG!");
}
```

출력 결과:

```
Listener2.SomethingHappend : Nuclear launch detected.
Listener3.SomethingHappend : Nuclear launch detected.

Listener1.SomethingHappend : Fire!!
Listener2.SomethingHappend : Fire!!

Listener1.SomethingHappend : RPG!
```

익명 메서드

- 이름이 없는 메서드
  - ✓ delegate 키워드를 사용해 선언하고 대리자 인스턴스에 연결 가능
  - ✓ 익명 메서드는 자신을 참조할 대리자와 동일한 반환/매개변수 형식 사용

사용 형식:

```
대리자_인스턴스 = delegate ( 매개변수_목록 )
                  {
                      // 실행하고자 하는 코드 ...
                  }
```

사용 예제:

```
delegate int Calculate(int a, int b);

static void Main(string[] args)
{
    Calculate Calc;
    Calc = delegate (int a, int b) // 대리자 인스턴스의 익명 메서드 참조
           {
               return a + b;
           };

    Console.WriteLine($"3 + 4 : {Calc(3,4)}"); // 출력: 3 + 4 : 7
}
```

익명 메서드 예제 코드

```
delegate int Compare(int a, int b);

static void BubbleSort(int[] DataSet, Compare Comparer)
{
    int i = 0;
    int j = 0;
    int temp = 0;

    for (i = 0; i < DataSet.Length - 1; i++)
    {
        for (j = 0; j < DataSet.Length - (i + 1); j++)
        {
            if (Comparer(DataSet[j], DataSet[j + 1]) > 0)
            {
                temp = DataSet[j + 1];
                DataSet[j + 1] = DataSet[j];
                DataSet[j] = temp;
            }
        }
    }
}
```

```
static void Main(string[] args)
{
    int[] array = { 3, 7, 4, 2, 10 };

    Console.WriteLine("Sorting ascending...");
    BubbleSort(array, delegate(int a, int b)
    {
        if (a > b)
            return 1;
        else if (a == b)
            return 0;
        else
            return -1;
    });

    for (int i = 0; i < array.Length; i++)
        Console.Write($"{array[i]} ");
}
```

출력 결과:

```
Sorting ascending...
2 3 4 7 10
```

익명 메서드 예제 코드

```
delegate int Compare(int a, int b);

static void BubbleSort(int[] DataSet, Compare Comparer)
{
    int i = 0;
    int j = 0;
    int temp = 0;

    for (i = 0; i < DataSet.Length - 1; i++)
    {
        for (j = 0; j < DataSet.Length - (i + 1); j++)
        {
            if (Comparer(DataSet[j], DataSet[j + 1]) > 0)
            {
                temp = DataSet[j + 1];
                DataSet[j + 1] = DataSet[j];
                DataSet[j] = temp;
            }
        }
    }
}
```

```
static void Main(string[] args)
{
    int[] array2 = { 7, 2, 8, 10, 11 };
    Console.WriteLine("₩nSorting descending...");
    BubbleSort(array2, delegate (int a, int b)
    {
        if (a < b)
            return 1;
        else if (a == b)
            return 0;
        else
            return -1;
    });

    for (int i = 0; i < array2.Length; i++)
        Console.Write($"{array2[i]} ");
}
```

출력 결과:

```
Sorting descending...
11 10 8 7 2
```

이벤트

- WinForm에서 버튼 클릭, 콤보 박스에서 값 선택 등 발생한 사건을 알리는 것을 의미
- 이벤트는 대리자를 event 한정자로 수식해서 생성
  - ✓ 이벤트를 선언하고 사용하는 방법을 예제를 통해 보면,

Step 1: 대리자를 선언. 클래스 밖에 선언해도 되고 안에 선언해도 됨.

```
delegate void EventHandler(string message);
```

Step 2: 클래스 내에 Step 1에서 선언한 대리자의 인스턴스를 event 한정자로 수식해서 선언

```
class MyNotifier
{
    public event EventHandler SomethingHappened;

    public void DoSomething(int number)
    {
        int temp = number % 10;

        if(temp != 0 && temp % 3 == 0) // number가 3, 6, 9로 끝나는 값이
        {                              // 될 때마다 이벤트가 발생
            SomethingHappened(String.Format($"{number} : 짝"));
        }
    }
}
```

EventHandler는 Step 1 절차에서 선언한 대리자

이벤트

- 이벤트는 대리자를 event 한정자로 수식해서 생성
  - ✓ 이벤트를 선언하고 사용하는 방법을 예제를 통해 보면,

Step 3: 이벤트 핸들러 작성. 이벤트 핸들러는 Step 1에서 선언한 대리자와 일치하는 메서드

```
static public void MyHandler(string message)
{
    Console.WriteLine(message);
}
```

SomethingHappened 이벤트에서 사용할 이벤트 핸들러(MyHandler)는 EventHandler 대리자의 형식과 동일한 메소드이여야 함

Step 4: 이벤트가 존재하는 클래스의 인스턴스를 생성하고, 이벤트에 Step 3에서 작성한 이벤트 핸들러를 등록

```
static void Main(string[] args)
{
    MyNotifier notifier = new MyNotifier();
    notifier.SomethingHappened += new EventHandler(MyHandler);

    for(int i=1;i<30; i++)
    {
        notifier.DoSomething(i); // 이벤트가 발생하면 이벤트 핸들러가 호출
    }
}
```

SomethingHappened 이벤트에 MyHandler() 메소드를 이벤트 핸들러로 등록

이벤트 예제 코드

```
delegate void EventHandler(string message);

class MyNotifier
{
    public event EventHandler SomethingHappened;

    public void DoSomething(int number)
    {
        int temp = number % 10;

        if (temp != 0 && temp % 3 == 0)
        {
            SomethingHappened(String.Format($"{number} : 짝"));
        }
    }
}
```

출력 결과:

```
3 : 짝
6 : 짝
9 : 짝
13 : 짝
16 : 짝
19 : 짝
23 : 짝
26 : 짝
29 : 짝
```

```
static public void MyHandler(string message)
{
    Console.WriteLine(message);
}

static void Main(string[] args)
{
    MyNotifier notifier = new MyNotifier();
    notifier.SomethingHappened += new EventHandler(MyHandler);

    for(int i=1;i<30; i++)
    {
        notifier.DoSomething(i);
    }
}
```

대리자와 이벤트

- 이벤트는 단지 대리자를 event 키워드로 수식해서 선언한 것을 의미
- 대리자와 다른 이벤트의 특징
  - ✓ 이벤트가 선언된 클래스 밖에서 직접 이벤트 호출 불가 (컴파일 에러 발생)
  - ✓ 단, 클래스 밖에서 해당 이벤트에 이벤트 핸들러 등록 또는 해지는 가능

```
static void Main(string[] args)
{
    MyNotifier notifier = new MyNotifier();
    notifier.SomethingHappened += new EventHandler(MyHandler);

    for(int i=1;i<30; i++)
    {
        notifier.DoSomething(i);
        notifier.SomethingHappened("테스트"); // 컴파일에러 발생.
    }
}
```

SomethingHappened 이벤트를
notifier 인스턴스의 클래스인
MyNotifier 밖에서 직접 호출 불가

.NET에서 지원하는 이벤트 처리

- .NET에서 이미 정의된 이벤트 대리자 EventHandler 사용하여 이벤트 생성 가능
  - ✓ 클래스에서 EventHandler 를 이용해 이벤트 선언
  - ✓ 클래스 외부에서 자유롭게 이벤트를 처리하기 위한 메서드 등록 및 해지 가능
  - ✓ 주의 할 점은 이벤트 발생은 오직 이벤트가 선언된 클래스 내부에서만 가능
  - ✓ 이벤트 대리자 EventHandler의 첫 번째 매개변수는 이벤트를 발생시킨 타입의 인스턴스이고, 두 번째 매개변수는 .NET에서 이미 정의된 System.EventArgs 타입의 이벤트에 속한 값

C#에서 이미 정의된 이벤트 처리를 위한 대리자 EventHandler

```
public delegate void EventHandler(object? sender, EventArgs e);
```

예제 코드

```
class PrimeCallbackArg : EventArgs
{// 이벤트에 속한 값을 담는 클래스 정의
    public int Prime;

    public PrimeCallbackArg(int prime)
    {
        this.Prime = prime;
    }
}
```

```
class PrimeGenerator
{   // C#에서 정의된 EventHandler를 이용해 이벤트 선언
    public event EventHandler PrimeGenerated;

    public void Run(int limit)
    {
        for(int i=2; i<= limit; i++)
        {// 소수의 경우 이벤트 발생
            if (IsPrime(i) == true && PrimeGenerated != null)
                PrimeGenerated(this, new PrimeCallbackArg(i));
        }
    }

    private bool IsPrime(int candidate)
    {// 매개변수로 입력 받은 정수 candidate이 소수인지 판별
        if ((candidate & 1) == 0)
            return candidate == 2;
        for (int i = 3; (i * i) <= candidate; i += 2)
        {
            if ((candidate & i) == 0) return false;
        }
        return candidate != 1;
    }
}
```

```
class MainApp
{
    static void PrintPrime(object sender, EventArgs arg)
                                                    .Prime + ", ");
                                                    gs arg)
                                                    ();
}
```

예제 코드

```
class PrimeCallbackArg : EventArgs
{
    public int Prime;

    public PrimeCallbackArg(int prime)
    {
        this.Prime = prime;
    }
}
```

```
class PrimeGenera
{
    public event

    public void P
    {
        for(int
        {
            if (
                P
        }
    }

    private bool
    {
        if ((cand
            retu
        for (int
        {
            if (
        }
```

출력 결과:

```
2, 3, 5, 7, 9,
26
2, 3, 5, 7, 9, 11, 13, 15,
```

```
class MainApp
{
    static void PrintPrime(object sender, EventArgs arg)
    { // 이벤트를 처리하기 위한 메소드 1
        Console.Write((arg as PrimeCallbackArg).Prime + ", ");
    }

    static int Sum;

    static void SumPrime(object sender, EventArgs arg)
    { // 이벤트를 처리하기 위한 메소드 2
        Sum += (arg as PrimeCallbackArg).Prime;
    }

    static void Main(string[] args)
    {
        PrimeGenerator gen = new PrimeGenerator();

        gen.PrimeGenerated += PrintPrime;
        gen.PrimeGenerated += SumPrime;

        gen.Run(10);
        Console.WriteLine();
        Console.WriteLine(Sum);

        gen.PrimeGenerated -= SumPrime;
        gen.Run(15);
    }
}
```

Thank you!